N95-19749

A Parallel Strategy for Implementing Real-Time Expert Systems Using CLIPS

Laszlo A. Ilyes & F. Eugenio Villaseca
Cleveland State University
Dept. of Electrical Engineering
1983 East 24th Street
Cleveland, Ohio 44115

John DeLaat
NASA Lewis Research Center
21000 Brookpark Road
Cleveland, Ohio 44135

ABSTRACT

As evidenced by current literature, there appears to be a continued interest in the study of real-time expert systems. It is generally recognized that speed of execution is only one consideration when designing an effective real-time expert system. Some other features one must consider are the expert system's ability to perform temporal reasoning, handle interrupts, prioritize data, contend with data uncertainty, and perform context focusing as dictated by the incoming data to the expert system.

This paper presents a strategy for implementing a real time expert system on the iPSC/860 hypercube parallel computer using CLIPS. The strategy takes into consideration, not only the execution time of the software, but also those features which define a true real-time expert system. The methodology is then demonstrated using a practical implementation of an expert system which performs diagnostics on the Space Shuttle Main Engine (SSME).

This particular implementation uses an eight node hypercube to process ten sensor measurements in order to simultaneously diagnose five different failure modes within the SSME. The main program is written in ANSI C and embeds CLIPS to better facilitate and debug the rule based expert system.

INTRODUCTION

Strictly defined, an expert system is a computer program which imitates the functions of a human expert in a particular field [1]. An expert system may be described as a real-time expert system if it can respond to user inputs within some reasonable span of time during which input data remains valid. A vast body of recently published research clearly indicates an active interest in the area of real-time expert systems [2-12].

Science and engineering objectives for future NASA missions require an increased level of autonomy for both onboard and ground based systems due to the extraordinary quantities of information to be processed as well as the long transmission delays inherent to space missions. [13]. An expert system for REusable Rocket Engine Diagnostics Systems (REREDS) has been investigated by NASA Lewis Research Center [14, 15, 16]. Sequential implementations of the expert system have been found to be too slow to analyze data for practical implementation. As implemented sequentially, REREDS already exhibits a certain degree of inherent parallelism. Ten

```
                    ( Value  "How are you ?" )
                )
    )
    ; Send the buffer to our new friend?
    ( BMSend ?Identity ?BufRef )

    ; Destroy the buffer
    ( BMDestroy ?BufRef )
)
```

```
(defrule RECEIVE
    (declare (salience -10000 ) )
    ?REC <- ( RECEIVE )
=>
    ( BMReceive )
    ( retract ?REC )
    ( assert( RECEIVE ) )
)
```

Figure 5. A CMS Sample

CONCLUSION

PVM and CLIPS both provide free source code systems that are well maintained by developers and a sizable number of users. Relative few source code changes are necessary to either system in order to build a reliable and robust platform that will support distributed computing in a heterogeneous environment of CPUs operating under UNIX. The CMS system described in this paper provides the CLIPS interface code and some parsing code sufficient to enable efficient use of PVM facilities and communication of CLIPS facts and templates among C, C^{++}, and CLIPS processes within a PVM virtual machine. Even more efficient communication can be obtained through enhancements to the PVM source code that can provide more efficient allocation of memory and reuse of PVM message buffers in certain applications.

NOTES

Information on PVM is best obtained by anonymous ftp from: netlib2.cs.utk.edu
Shar and tar packages are available from the same source.

The authors are currently using the CMS system in applications that involve multiple CLIPS expert systems in sophisticated interactive user interface settings. It is expected that the basic CMS code will become available in the Spring, 1995. Inquiries via e-mail are preferred.

BIBLIOGRAPHY

1. Riley, Gary, B. Donnell et. al., "CLIPS Reference Manual," JSC-25012, Lyndon B. Johnson Space Center, Houston, Texas, June, 1993.

2. Pohl, Jens, A. Chapman, L.Chirica, R. Howell, and L. Myers, "Implementation Strategies for a Prototype ICADS Working Model," CADRU-02-88, CAD Research Unit, Design Institute, School of Architecture and Design, Cal Poly, San Luis Obispo, California, June, 1988.

3. Myers, Leonard and J. Pohl, "ICADS Expert Design Advisor: An Aid to Reflective Thinking," Knowledge-Based Systems, London, Vol. 5, No. 1, March, 1992, pp. 41-54.

4. Pohl, Jens and L. Myers, "A Distributed Cooperative Model for Architectural Design," Automation In Construction, Amsterdam, 3, 1994, pp. 177-185.

6. Geist, Al, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, "PVM 3 User's Guide and Reference Manual," ORNL/TM-12187, Oak Ridge National Laboratory, Oak Ridge . Tennessee, May, 1993.

sensor measurements are used to diagnose the presence of five different failures which may manifest themselves in the working SSME. Each module of code which diagnoses one failure is referred to as a failure detector. While some of the sensor measurements are shared between failure detectors, the computations within these detectors are completely independent of one other.

One apparent way to partition the problem of detecting failures in the SSME, is to assign each failure detector to its own node on the hypercube system. Because the failure detectors may be processed simultaneously, a speedup in the execution is expected. But while execution time is a critical parameter in any real-time expert system, it is not the only ingredient required in order to guarantee its success. A recent report characterized the features required of expert systems to operate in real-time. In addition to the requirement of fast execution, the real-time expert system should also possess the ability to perform context focusing, interrupt handling, temporal reasoning, uncertainty handling, and truth maintenance. Furthermore, the computational time required by the system should be predictable and the expert system should potentially be able to communicate with other expert systems [17]. These aspects are considered in the design presented in this paper.

METHODS

The rules for diagnosing failures in the SSME were elicited from NASA engineers and translated into an off-line implementation of a REREDS expert system [18]. While some of the failures can be diagnosed using only sensor measurements, other failures require both data measurements and the results obtained from condition monitors. The condition monitors measure both angular velocity and acceleration on various bearings of the High Pressure Oxidizer Turbo-Pump (HPOTP) shaft and determine the magnitudes of various torsional modes in the HPOTP shaft [19]. Due to the lack of availability of high frequency bearing data and additional hardware requirements for implementing real-time condition monitors, this expert system considered only those failure detectors which required sensor measurements alone.

The five failure detectors which rely solely on sensor measurements for diagnosis are listed in Table 1 along with a description of the failure, the required sensor measurements, and their respective, relative failure states. Notice that failure detectors designated F11 and F15 cannot be differentiated from one another and are thus combined into one single failure mode.

For each sensor measurement listed, the expert system knowledge base is programmed with a set of nominal values and deviation values (designated in our work by σ). One of the roles of the expert system is to match incoming sensor measurements with the nominal and deviation values which correspond to the specific power level of the SSME at any given time. Any sensor measurement may deviate from the nominal value by $\pm\ \sigma$ without being considered high or low relative to nominal. Beyond the σ deviation, the sensor measurement is rated with a value which is linearly dependent upon the amount of deviation. This value is referred to as a vote and is used by a failure detector to determine a confidence level that the failure mode is present. This voting curve is illustrated in Figure 1.

Once a vote has been assigned to every sensor measurement, each failure detector averages the votes for all of its corresponding sensor measurements. The final result will be a number between -1.00 and +1.00. This result is converted to a percent and is

Failure Detector	Description	Measurements	Failure States
F11/15	Labyrinth/Turbine Seal Leak	LPFP Discharge Pressure	Low
		FPOV Valve Position	High
		HPFTP Turbine Discharge Temp.	High
F67	HPOTP Turbine Interstage & Tip Seal Wear	HPOTP Discharge Temperature	Low
		HPOP Discharge Pressure	Low
		HPOTP Shaft Speed	Low
		MCC Pressure	Low
F68	Intermediate Seal Wear	Secondary Seal Drain Temperature	Low
		HPOTP Inter-Seal Drain Pressure	Low
F69	HPOP Primary Seal Wear	HPOP Primary Seal Drain Pressure	High
		Secondary Seal Drain Pressure	High
		Secondary Seal Drain Temperature	High
F70	Pump Cavitation	HPOP Discharge Pressure	Low
		HPOTP Shaft Speed	Low
		MCC Pressure	Low

Table 1. - Failure Detectors Only Requiring Sensor Measurements for Failure Diagnosis

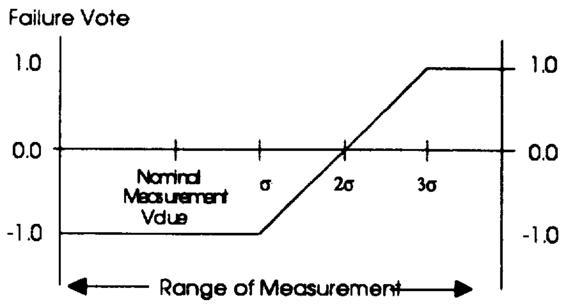

Figure 1. - Voting Curve for Sensor Measurement with "High" Failure State

referred to as the corresponding confidence level of that failure mode. The underlying motivation for this approach is to add inherent uncertainty handling to the expert system.

Each individual failure detector was implemented in CLIPS on a personal computer and its accuracy was tested and verified using simulated SSME sensor data. Once satisfactory results were achieved, an ANSI C program was written for the iPSC/860 hypercube computer which would initialize the CLIPS environment on five nodes of a 2^3 hypercube structure. These five nodes, referred to as the *failure detector nodes*, load the constructs for one failure detector each, and use CLIPS as an embedded application as described in the CLIPS Advanced Programming Guide [20]. In this way, CLIPS will only be used for evaluation of the REREDS rules. All other programming requirements, including opening and closing of sensor measurement data files, preliminary data analysis, and program flow control are handled in C language. By embedding the CLIPS modules within ANSI C code, context focusing and process interruptions can be more efficiently realized.

Coordination of data acquisition and distribution among the failure detector nodes is accomplished through a *server node* which is programmed to furnish sensor measurement data to requesting nodes. Since the data for this study originate from the SSME simulator test bed, data retrieval is accomplished simply by reading sequential data from prepared data files. The server node transfers incoming sensor measurements into an indexed memory array, or blackboard, from which data are distributed upon request to the failure detector nodes. When the blackboard is updated, all requests for data are ignored until data transfer is completed. This assures that reasoning within the expert system is always performed on contemporaneous data. The server node does not invoke the CLIPS environment at any time. It is programmed entirely in C language code.

One additional node, referred to as the *manager node*, is used by the expert system to coordinate the timing between the failure detector nodes and the server node. Like the server node, the manager node does not invoke the CLIPS environment. Once the manager node has received a "ready" message from all failure detector nodes, it orders the server node to refresh the data in the blackboard. During this refresh, the failure detector nodes save their results to permanent storage on the system. The activities and process flow of all three types of nodes used in this research are illustrated in Figure 2. The asterisk denotes the point at which all nodes synchronize.

CONCLUSION

Profiling studies were conducted on the parallel implementation of the REREDS expert system. It was found that the system could process the sensor measurements and report confidence levels for all five failure modes in 18 milliseconds. A sequential implementation of the expert system on the same hardware was found to require over 50 milliseconds to process and report the same information, indicating that the parallel implementation can process data at nearly three times as quickly. Considering the fact that seven processors are being used in the parallel implementation, these results may seem somewhat disappointing, however, the profiling studies also indicate that additional speedup can be realized in future implementations of this expert system if the data blackboard is also parallelized. Using only one server node causes some hardware contention. Shortly after the nodes synchronize, the failure detectors tend to overwhelm the server with five

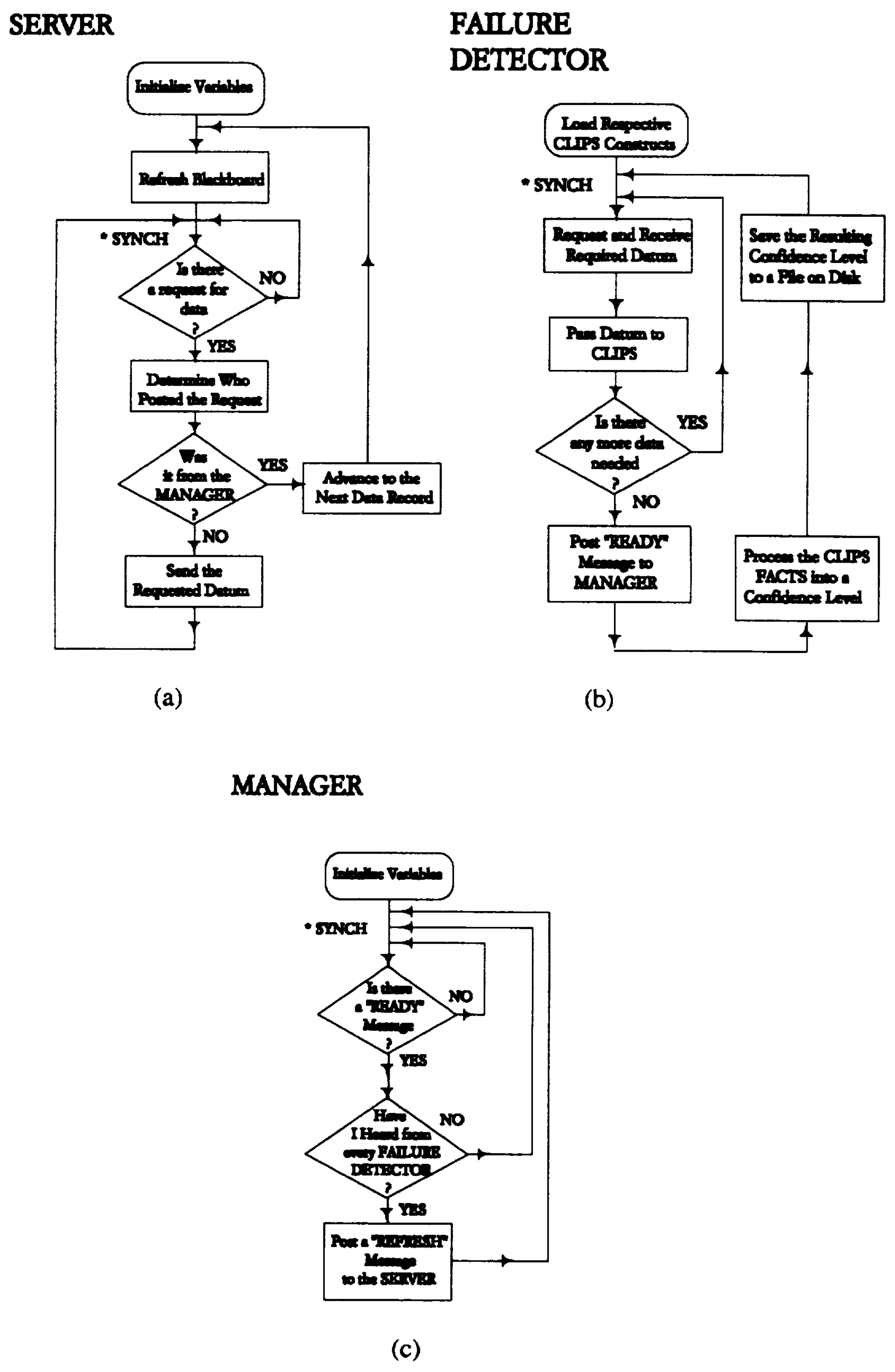
SERVER
Initialize Variables
Refresh Blackboard
* SYNCH
Is there a request for data ?
NO
YES
Determine Who Posted the Request
Was it from the MANAGER ?
YES
Advance to the Next Data Record
NO
Send the Requested Datum
(a)
FAILURE DETECTOR
Load Respective CLIPS Constructs
* SYNCH
Request and Receive Required Datum
Pass Datum to CLIPS
Is there any more data needed ?
YES
NO
Post "READY" Message to MANAGER
Save the Resulting Confidence Level to a File on Disk
Process the CLIPS FACTS into a Confidence Level
(b)
MANAGER
Initialize Variables
* SYNCH
Is there a "READY" Message ?
NO
YES
Have I Heard from every FAILURE DETECTOR ?
NO
YES
Post a "REFRESH" Message to the SERVER
(c)

Figure 2. - Process flow for the a) Server Node, b) Failure Detector Nodes, and c) Manager Node

(nearly) simultaneous data requests. By adding a second server node to the system, this contention can be greatly reduced.

Since the data can be processed at a fast, continuous rate, the validity of sensor measurements can be assured during processing. Consequently, truth maintenance is realized by suppressing data requests to the server node until all sensor measurements have been simultaneously updated. This guarantees that all data accessed by the failure detector nodes during any processing cycle is the same "age."

Due to the nature of the particular expert system selected for this research, the time required by the failure detectors to process SSME data remains constant regardless of whether or not a failure condition exists. Thus, predictability is always assured for this example. Also, the need for temporal reasoning is not explicitly indicated and is therefore not investigated. Since these aspects of the design are application specific, they and must be investigated in future work using different expert system models.

As discussed earlier, uncertainty handling is inherent to this expert system. The voting scheme and use of confidence levels permits reasoning, even in the presence of noisy, incomplete, or inconsistent data. Since the output from the system is a graded value rather than a binary value, the output carries with it additional information about the expert system's confidence that a particular failure is occurring.

One of the most important features of this design is that program flow control and system I/O is accomplished in C language code. Using CLIPS as an embedded application within a fast, compiled body of C language code allows the expert system to be more easily integrated into a practical production system. Complex reasoning can be relegated directly and exclusively to the nodes invoking the CLIPS environment, while tasks which are better suited to C language code can be performed by the server and manager nodes. Thus, simple decisions can be realized quickly in C language rather than relying on the slower CLIPS environment. Based on fast preprocessing of the sensor measurements, the C language code can be used to initiate process interrupts during emergency conditions and even change the context focusing of the expert system. Those tasks which require complex reasoning can be developed and refined separately in CLIPS, taking full advantage of the debugging tools available in the CLIPS development environment.

While the rules for this particular expert system are somewhat simple compared to other applications considered in the literature, it is believed that the approach used in this study can be extended to other examples. This study demonstrates that parallel processing can not only speed up the execution of certain expert systems, but also incorporate other important features essential for real-time operation.

ACKNOWLEDGMENTS

We would like to thank NASA Lewis Research Center for its financial support of this research under cooperative agreement NCC-257 (supplement no. 3). We would also like to extend our thanks to the technical support staff of the ACCL and the internal reviewers at NASA Lewis

Research Center, as well as Mr. Gary Riley of NASA Johnson Space Center. It is only with their patient and supportive contributions that this research was possible.

REFERENCES

1. Lugger, George and Stubblefield, William, "AI: History and Applications," ARTIFICIAL INTELLIGENCE AND THE DESIGN OF EXPERT SYSTEMS, The Benjamin/Cummings Publishing Co. Inc., New York, NY, 1989, pp. 16-19.

2. Leitch, R., "A Real-Time Knowledge Based System for Process Control," IEE PROCEEDINGS, PART D: CONTROL, THEORY AND APPLICATIONS, Vol. 138, No. 3, May, 1991, pp. 217-227.

3. Koyama, K.,"Real-Time Expert System for Gas Plant Operation Support (GAPOS)," 16TH ANNUAL CONFERENCE OF IEEE INDUSTRIAL ELECTRONICS, November, 1990.

4. Bahr, E., Barachini, F., Dopplebauer, J. Grabner, H. Kasperic, F., Mandl, T., and Mistleberger, H., "Execution of Real-Time Expert Systems on a Multicomputer," 16th ANNUAL CONFERENCE OF IEEE INDUSTRIAL ELECTRONICS, November, 1990.

5. Wang, C. Mok, A., and Cheng, A., "A Real-Time Rule Based Production System," PROCEEDINGS OF THE 11TH REAL-TIME SYSTEMS SYMPOSIUM, December, 1990.

6. Moore, R., Rosenhof, H., and Stanley, G., "Process Control Using Real-Time Expert System," PROCEEDINGS OF THE 11TH TRIENNIAL WORLD CONGRESS OF THE INTERNATIONAL FEDERATION OF AUTOMATIC CONTROL, August, 1990.

7. Lee, Y., Zhang, L., and Cameron, R., "ESC: An Expert Switching Control System," INTERNATIONAL CONFERENCE ON CONTROL, March, 1991.

8. Jones, A., Porter, B., Fripp, R., and Pallet, S., "Real-Time Expert System for Diagnostics and Closed Loop Control," PROCEEDINGS OF THE 5TH ANNUAL IEEE INTERNATIONAL SYMPOSIUM ON INTELLIGENT CONTROL, September, 1990.

9. Borsje, H., Finn, G., and Christian, J., "Real-Time Expert Systems and Data Reconciliation for Process Applications," PROCEEDINGS OF THE ISA 1990 INTERNATIONAL CONFERENCE AND EXHIBITION, Part 4 of 4, October, 1990.

10. Silagi, R. and Friedman, P., "Telemetry Ground Station Data Servers for Real-Time Expert Systems," INTERNATIONAL TELEMETERING CONFERENCE, ITC/USA, October, 1990.

11. Spinrad, M., "Facilities for Closed-Loop Control in Real-Time Expert Systems for Industrial Automation," PROCEEDINGS OF THE ISA/1989 INTERNATIONAL CONFERENCE AND EXHIBIT, Part 2 of 4, October, 1990.

12. Hota, K., Nomura, H., Takemoto, H., Suzuki, K., Nakamura, S., and Fukui, S., "Implementation of a Real-Time Expert System for a Restoration Guide in a Dispatching Center," IEEE TRANSACTIONS ON POWER SYSTEMS, Vol. 5, No. 3, 1990, pp. 1032-8.

13. Lau, S. and Yan, J. , "Parallel Processing and Expert Systems," NASA Technical Memorandum, No. 103886, May, 1991.

14. Merrill, Walter and Lorenzo, Carl, "A Reusable Rocket Engine Intelligent Control," 24th Joint Propulsion Conference, Cosponsored by AIAA, ASME, and ASEE, Boston MA, July 1988.

15. Guo T.-H. and Merrill, W., "A Framework for Real-Time Engine Diagnostics," 1990 Conference for Advanced Earth-to-Orbit Propulsion Technology," Marshall Space Flight Center, May 1990.

16. Guo, T.-H., Merrill, W. , and Duyar, A., "Real-Time Diagnostics for a Reusable Rocket Engine," NASA Technical Memorandum No. 105792, August 1992.

17. Kreidler, D. and Vickers, D., "Distributed Systems Status and Control," Final Report to NASA Johnson Space Center, NASA CR 187014, September 1990.

18. Anex, Robert, "Reusable Rocket Engine Diagnostic System Design,"Final Report, NASA CR 191146, January 1993.

19. Randall, M.R., Barkadourian, S., Collins, J.J., and Martinez, C., "Condition Monitoring Instrumentation for Space Vehicle Propulsion Systems," NASA CP 3012, pp. 562-569, May 1988.

20. CLIPS 6.0 User's Manual, Volume II, Advanced Programming Guide, pp. 43-142, June, 1993.